Apple Assembly Line

$1.50

Volume 3 -- Issue 8 May, 1983

In This Issue...

Yet Another Cross Assembler: PDP-11

We are turning the tables at last. When the 6502 was created six or seven years ago, programmers used PDP-11 development systems with cross assemblers to write 6502 code. Now you can use your Apple to write programs for the Digital Equipment Corporation's -11 family. Thanks to Martin Buchholtz for encouraging us to develop this one. He plans to use it for writing programs to run in DEC Falcon SBC-11 based systems. Only $50, if you already own the S-C Macro Assembler. See our ad on page 16 for more about the Cross Assemblers.

We Need Your Help

Does anybody have complete details of the file format of the Apple ///'s relocatable object files? That's the last remaining stumbling block on the road to the S-C Macro Assembler ///. Has anyone figured it out yet?

All Around the World

We are now sending the Apple Assembly Line to subscribers in 32 different countries. (That's about 1200 copies to the USA, and about 100 copies to the other 31 nations.)

Displaying Character Generator EPROMs.....Bob Sander-Cederlof

We make our own Character Generator EPROMs for Revision 7 or later Apple II Plusses. I use the Mountain Hardware EPROM Burner to burn the data into 2716 EPROMs. We have several different character sets, and it can be a lot of trouble to check the results.

After designing a character set, and formatting all the bits into the 2048 bytes of EPROM space, and burning it in, we still have to take an Apple apart and plug the chip in to see if all the characters look right.

I decided to write a program which would map the EPROM data onto the hi-res screen, allowing me to test without wasting time burning/erasing EPROMs and dismantling/re-assembling my Apple.

Even if you don't have the same requirements, you can learn a lot about indexing techniques and address shuffling from studying the following program.

Starting at the top.... I set up three page-zero variables in lines 1040-1060. The S-C Macro Assembler is a great environment for making short programs like this one, because I can cycle through edit-assemble-test until it works just right without ever leaving the assembler. S-C Macro allows me to use zero-page locations $00-$1F without fear of inteference ($00-$1E in the Apple //e).

Lines 1080 and 1090 define two buffers where I BLOAD two different EPROM images. I put one at $6800-6FFF, the other at $7000-77FF. There is room on the screen to display one character set in a 16x16 matrix on the left side, and the other on the right side.

For grins, I decided to use the subroutine in Applesoft ROM at $F3E2 to turn on hi-res mode. This is the code executed for the HGR statement, so I called it AS.HGR at line 1110. HGR sets all the soft-switches to hi-res page 1, and clears the screen.

Lines 1160-1180 call the HGR subroutine. Since I was using S-C Macro in the RAM card, and since the Applesoft ROMs are not switched on when a program is executing in the RAM card, I had a problem. The first time I tried to run DISPLAY, I left out lines 1160 and 1180. The result was a total disaster. Line 1170 did a JSR $F3E2 into the RAM card! I had to RESET and reboot the computer to get control again. Look out for these kinds of problems whenever you are trying to use code in both places at once.

Lines 1190-1280 set up the starting addresses to display the first character set on the left half of the screen. Lines 1290-1380 do the same job to show the second set on the right half-screen.

S-C Macro Assembler (the best there is!).....$80.00
S-C Macro Assembler Version 1.1 Update.....$12.50
For registered owners of the S-C Macro Assembler.
Source code of S-C Assembler II, Version 4.0, on disk.....$95.00
Fully commented, easy to understand and modify to your own tastes.
S-C Macro Assembler /// $100.00
Preliminary version. Call or write for details.

S-C Cross Reference Utility $20.00
S-C Cross Reference Utility with Complete Source Code.....$50.00

S-C Word Processor.....$50.00
As is, with fully commented source code. Needs S-C Macro Assembler.
Applesoft Source Code on Disk.....$50.00
Very heavily commented. Requires Applesoft and S-C Assembler.
ES-CAPE: Extended S-C Applesoft Program Editor.....$60.00

AAL Quarterly Disks.....each $15.00
Each disk contains all the source code from three issues of "Apple Assembly Line", to save you lots of typing and testing time.
QD#1: Oct-Dec 1980 QD#2: Jan-Mar 1981 QD#3: Apr-Jun 1981
QD#4: Jul-Sep 1981 QD#5: Oct-Dec 1981 QD#6: Jan-Mar 1982
QD#7: Apr-Jun 1982 QD#8: Jul-Sep 1982 QD#9: Oct-Dec 1982
QD#10: Jan-Mar 1983

Double Precision Floating Point for Applesoft.....$50.00
Provides 21-digit precision for Applesoft programs.
Includes sample Applesoft subroutines for standard math functions.

FLASH! Integer BASIC Compiler (Laumer Research)..... $79.00
Source Code for FLASH! Runtime Package.....$39.00
Full Screen Editor for S-C Macro Assembler (Laumer Research).....$49.00

The Visible Computer: 6502 (Software Masters).....(reg. $50.00) $45.00
Super Disk Copy III (Sensible Software).....(reg. $30.00) $27.00
Amper-Magic (Anthro-Digital).....(reg. $75.00) $67.50
Amper-Magic Volume 2 (Anthro-Digital).....(reg. $35.00) $30.00
Quick-Trace (Anthro-Digital).....(reg. $50.00) $45.00
DISASM Dis-Assembler (Rak-Ware).....$30.00

Blank Diskettes (with hub rings).....package of 20 for $50.00
Small 3-ring binder with 10 vinyl disk pages and disks.....$36.00
Vinyl disk pages, 6"x8.5", hold one disk each.....10 for $6.00
Reload your own NEC PC-8023 ribbon cartridges.....each ribbon $5.00
Reload your own NEC Spinwriter Multi-Strike Film cartridges....each $2.50
Diskette Mailing Protectors.....10-99: 40 cents each
100 or more: 25 cents each
ZIF Game Socket Extender.....$20.00
Ashby Shift-Key Mod.....$15.00
Lower-Case Display Encoder ROM.....$25.00
Only Revision level 7 or later Apples.

Grappler+ Printer Interface (Orange Micro).....($175.00) $150.00
Bufferboard 16K Buffer for Grappler (Orange Micro).....($175.00) $150.00
Buffered Grappler+ NEW!! Interface and 16K Buffer.....($239.00) $200.00

Books, Books, Books.....compare our discount prices!
"The Apple][Circuit Description", Gayler.....($22.95) $21.00
"Enhancing Your Apple II, vol. 1", Lancaster.....($15.95) $15.00
"Incredible Secret Money Machine", Lancaster.....($7.95) $7.50
"Micro Cookbook, vol. 1", Lancaster.....($15.95) $15.00
"Beneath Apple DOS", Worth & Lechner.....($19.95) $18.00
"Bag of Tricks", Worth & Lechner, with diskette.....($39.95) $36.00
"Apple Graphics & Arcade Game Design", Stanton.....($19.95) $18.00
"Assembly Lines: The Book", Roger Wagner.....($19.95) $18.00
"What's Where in the Apple", Second Edition.....($24.95) $23.00
"What's Where Guide" (updates first edition).....($9.95) $9.00
"6502 Assembly Language Programming", Leventhal.....($16.99) $16.00
"6502 Subroutines", Leventhal.....($12.99) $12.00

Add $1 per book for US postage. Foreign orders add postage needed.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
*** (214) 324-2050 ***
*** We take Master Charge, VISA and American Express ***

The top line of hi-res page 1 starts at $2000, and goes to $2027. The middle of the line starts at $2014. The starting addresses of subsequent lines can be computed from these two base addresses, although it is a little tricky. More on this later.

The hi-res screen shows the least significant seven bits from each byte. There are forty bytes in each line, making a total of 280 dots across. The dots in each byte are in reverse order: the least significant bit is the leftmost dot. On the other hand, the EPROM image is in normal order. The subroutine DISPLAY.ONE.SET takes care of all the addressing, and REVERSE.BITS handles the reversals.

Lines 1400-1410 pause until I hit any key on the keyboard. During this pause I can examine the screen as long as I wish. When I type any key, the keyboard strobe will be set and $C000 will go negative. Line 1420 will then clear the keyboard strobe, and the RTS at line 1430 returns to the S-C Macro Assembler.

This brings us to a closer examination of the subroutine to actually display a character set, in lines 1440-1770. We will be displaying 16 rows of characters, with 16 characters in each row. It is therefore natural to simplify the problem by writing another subroutine to display one row of characters, and call it sixteen times.

Lines 1480 and 1490 start a loop much like Applesoft's FOR I = 1 TO 16...except in assembly language it is easier to go from 16 to 1. The equivalent to NEXT I is at lines 1750 and 1760, where CNT16 is decremented. In between we have the body of the loop.

Line 1500 calls DISPLAY.ONE.ROW, a subroutine that only gets called from this one line. I made it into a separate subroutine so I could put off writing it until later, and concentrate on one loop at a time. DISPLAY.ONE.ROW expects the addresses at SCREEN.ADR and EPROM.ADR to be already set up for the first byte to be displayed in the current row. After it returns, those addresses will have been modified.

Lines 1510-1580 add 15*8, or 120, to the address in EPROM.ADR. DISPLAY.ONE.ROW already added 8, so the total augment is 128. This moves us up to the beginning of the next set of sixteen characters.

Lines 1590-1740 assume that DISPLAY.ONE.ROW already added $2000 to the address in SCREEN.ADR, and subtracts that value back out. At the same time, we add back in $80, to move to the next group of eight screen lines for the next row of characters. This is sufficient for the first eight rows of characters, but in moving to the ninth row there is a discontinuity which requires adding $28 and subtracting $400 to get the right address. The fact that the ninth row has arrived is apparent by the fact that the high byte of the address goes above $23 (lines 1670 and 1680).

Here is a table of the starting addresses for each of the 24 character rows (we only use the first 16):

Row	Address	Row	Address	Row	Address
1	$2000	9	$2028	17	$2050
2	$2080	10	$20A8	18	$20D0
3	$2100	11	$2128	19	$2150
4	$2180	12	$21A8	20	$21D0
5	$2200	13	$2228	21	$2250
6	$2280	14	$22A8	22	$22D0
7	$2300	15	$2328	23	$2350
8	$2380	16	$23A8	24	$23D0

The starting addresses for the right half-screen can be obtained by just adding $14 to all of the above addresses. What we do is START at $2014, and all the rest are computed automatically.

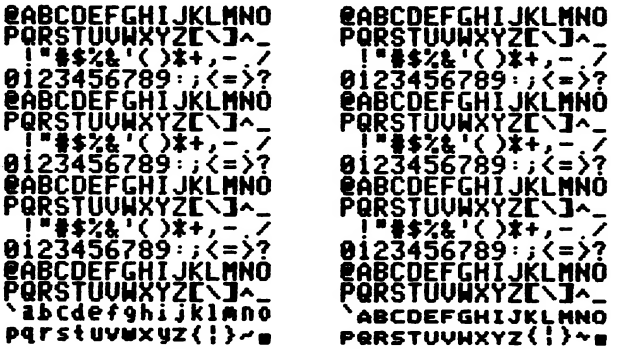
Now we can talk about what goes on inside one row of characters. Lines 1810-2000 do the job of moving bytes from the EPROM image to the eight screen lines which form the row of characters. Lines 1820-1830 start a loop to count out eight repetitions, and lines 1980-1990 perform the NEXT on this loop.

On each pass through the loop the subroutine GET.PUT is called sixteen times to move a byte for each character to the screen image. GET.PUT is another subroutine only called from one place, but made into a subroutine for ease of understanding. The inner loop of 0 through 15 is controlled by the X-register. Line 1850 sets X=0, and lines 1910-1930 increment, test, and branch ("NEXT X" sequence). The X-register also indexes the STA instruction inside GET.PUT, so that the screen byte for each character is stored into the right place on the screen line. The Y-register is used as an index into the EPROM data by GET.PUT, and parallels the X-register but with an increment of 8 rather than 1. Lines 1870-1900 bump the Y-register by 8 each time through the inner loop.

GET.PUT (lines 2230-2340) does the very simple job of moving one byte from one place in memory to another. Or is it so simple.... Notice that the addresses inside the LDA and STA instructions are filled in when the program runs. This is called self-modifying code, and I normally avoid such code at all costs. It can lead to all sorts of devastating things. Nevertheless, there are exceptions to most rules, and a time for nearly everything. This is one of those, I think. Isolating the offensive code into its own little subroutine appeases my conscience somewhat.

In between LDA and STA I call REVERSE.BITS, yet another simple subroutine which could be written in-line. I prefer making it separate for nicer modularity. The comments show what is going on, bit-by-bit. If you were working from character generator data written for the DOS TOOL KIT or HIGHER TEXT, the bits would already be in the right order. It is just because I am using data for the character generator EPROM that we need to reverse the bits.

Here is a printout done with my NEC PC-8023 and a Grappler+ interface card. The two character sets shown are the ones we sell. The one on the left uses regular lower case characters, with descenders. All the lower case characters are raised up one screen line to leave room for the descenders. The set on the right uses small caps for the lower case, and is the one we use in all the Apples here. The first four rows are the characters used in INVERSE mode, and the next four rows are for FLASH mode. (Doesn't flash too well on paper!)



FASTDRAW 1.1

A software package from CASTLE DESIGNS which provides enhanced Hi-Resolution Graphics instructions for use in Applesoft Basic programs. For example ....

&DRAW,1	Erases, moves and re-draws an entire array of shapes with a single instruction.
&HPLOT,1	Erases, moves and replots an array of points.
&HGR,c	Performs a hi-speed screen erase to any color.

Plotting rate exceeds 4,000 points per second.

Other FASTDRAW 1.1 instructions provide scrolling, as well as drawing or plotting without erasing. Assembly language entry points are also provided. These new commands may be used to create fast-moving Basic games or to plot data quickly.

- - - - - - - - - - - -

This software may be used with the APPLIED ENGINEERING A/D BOARD to acquire and plot analog data. The A/D BOARD will acquire 8 channels of data with 8-bit resolution, and is very easy to use. Simply use a PEEK statement to read the data.

APPLIED ENGINEERING A/D BOARD $129.00

To acquire data with real-time accuracy, use the APPLIED ENGINEERING TIMEMASTER real time clock board in combination with the A/D BOARD and the FASTDRAW 1.1 software. The TIMEMASTER will provide timing for data acquisition as often as once per millisecond. For example, 200 data points can be acquired and plotted in 250 milliseconds.

APPLIED ENGINEERING TIMEMASTER $129.00

- - - - - - - - - - - -

Included on the FASTDRAW 1.1 diskette are several programs which demonstrate its various uses and a utility which merges two shape tables. Special routines in both Basic and assembly language are provided which utilize the APPLIED ENGINEERING boards to acquire and plot data. Manual included. Requires Applesoft and DOS 3.3, and an APPLE II, II+, or IIe w/48K Ram.

CASTLE DESIGNS FASTDRAW 1.1 ..... $29.95

- - - - - - - - - - - -

Texas Residents Add 5% Sales Tax (Hardware Only).
Add $10.00 If Outside U.S.A.

Order any of the above items from APPLIED ENGINEERING, P.O. BOX 470301, DALLAS, TEXAS 75247 ; Phone (214) 492-2027

Or order FASTDRAW 1.1 from CASTLE DESIGNS, 2717 TEAKWOOD, PLANO, TEXAS 75075

FASTDRAW 1.1

```
                        1010 *-------------------------------
                        1020 *       DISPLAY CHARACTER SET
                        1030 *-------------------------------
0000-                   1040 CNT8        .EQ $00
0001-                   1050 B           .EQ $01
0002-                   1060 CNT16       .EQ $02
                        1070 *-------------------------------
6800-                   1080 EPROM.A.IMAGE   .EQ $6800
7000-                   1090 EPROM.B.IMAGE   .EQ $7000
                        1100 *-------------------------------
F3E2-                   1110 AS.HGR      .EQ $F3E2
                        1120 *-------------------------------
                        1130         .OR $803
                        1140 DISPLAY
                        1150 *---TURN ON HI-RES GRAPHICS------
0803- AD 81 C0          1160         LDA $C081    GET A/S ROMS ON MOTHERBOARD
0806- 20 E2 F3          1170         JSR AS.HGR
0809- AD 80 C0          1180         LDA $C080    BACK TO S-C ASM IN RAM CARD
                        1190 *---FIRST CHAR SET---------------
080C- A9 20             1200         LDA /$2000   TOP LINE, LEFT SIDE
080E- 8D CC 08          1210         STA SCREEN.ADR+1
0811- A9 00             1220         LDA #$2000
0813- 8D CB 08          1230         STA SCREEN.ADR
0816- A9 68             1240         LDA /EPROM.A.IMAGE  FIRST CHARACTER SET
0818- 8D C6 08          1250         STA EPROM.ADR+1
081B- A9 00             1260         LDA #EPROM.A.IMAGE
081D- 8D C5 08          1270         STA EPROM.ADR
0820- 20 43 08          1280         JSR DISPLAY.ONE.SET
                        1290 *---SECOND CHAR SET--------------
0823- A9 20             1300         LDA /$2014   TOP LINE, RIGHT SIDE
0825- 8D CC 08          1310         STA SCREEN.ADR+1
0828- A9 14             1320         LDA #$2014
082A- 8D CB 08          1330         STA SCREEN.ADR
082D- A9 70             1340         LDA /EPROM.B.IMAGE  SECOND CHARACTER SET
082F- 8D C6 08          1350         STA EPROM.ADR+1
0832- A9 00             1360         LDA #EPROM.B.IMAGE
0834- 8D C5 08          1370         STA EPROM.ADR
0837- 20 43 08          1380         JSR DISPLAY.ONE.SET
                        1390 *---PAUSE UNTIL KEYSTROKE--------
083A- AD 00 C0          1400 .1      LDA $C000
083D- 10 FB             1410         BPL .1
083F- 8D 10 C0          1420         STA $C010
0842- 60                1430         RTS          RETURN TO ASSEMBLER
                        1440 *-------------------------------
                        1450 *       DISPLAY ONE CHARACTER SET IN 16-BY-16
                        1460 *-------------------------------
                        1470 DISPLAY.ONE.SET                          FORMAT
0843- A9 10             1480         LDA #16      COUNT 16 ROWS
0845- 85 02             1490         STA CNT16
0847- 20 85 08          1500 .1      JSR DISPLAY.ONE.ROW
                        1510 *---NEXT ROW IN EPROM DATA-------
084A- 18                1520         CLC
084B- AD C5 08          1530         LDA EPROM.ADR
084E- 69 78             1540         ADC #15*8
0850- 8D C5 08          1550         STA EPROM.ADR
0853- AD C6 08          1560         LDA EPROM.ADR+1
0856- 69 00             1570         ADC #0
0858- 8D C6 08          1580         STA EPROM.ADR+1
                        1590 *---NEXT ROW ON SCREEN-----------
085B- 38                1600         SEC
085C- AD CB 08          1610         LDA SCREEN.ADR
085F- E9 80             1620         SBC #$2000-$80
0861- 8D CB 08          1630         STA SCREEN.ADR
0864- AD CC 08          1640         LDA SCREEN.ADR+1
0867- E9 1F             1650         SBC /$2000-$80
0869- 8D CC 08          1660         STA SCREEN.ADR+1
086C- C9 24             1670         CMP #$24           HIT THE BREAK YET?
086E- 90 10             1680         BCC .2             NO, GO ON
0870- AD CB 08          1690         LDA SCREEN.ADR     YES, ADJUST THE
0873- E9 D8             1700         SBC #$400-$28                ADDRESSES
0875- 8D CB 08          1710         STA SCREEN.ADR
0878- AD CC 08          1720         LDA SCREEN.ADR+1
087B- E9 03             1730         SBC /$400-$28
087D- 8D CC 08          1740         STA SCREEN.ADR+1
0880- C6 02             1750 .2      DEC CNT16     LAST ROW YET?
0882- D0 C3             1760         BNE .1        ...NO
0884- 60                1770         RTS           ...YES, RETURN
```

```
                        1780 *-------------------------------
                        1790 *       DISPLAY ONE ROW OF 16 CHARACTERS
                        1800 *-------------------------------
                        1810 DISPLAY.ONE.ROW
0885- A9 08             1820         LDA #8          8 SCREEN LINES FOR ONE ROW
0887- 85 00             1830         STA CNT8
0889- A0 00             1840 .1      LDY #0          EPROM DATA INDEX
088B- A2 00             1850         LDX #0          SCREEN IMAGE INDEX
088D- 20 C4 08          1860 .2      JSR GET.PUT     MOVE ONE BYTE TO SCREEN
0890- 98                1870         TYA             ADD 8 TO EPROM DATA INDEX
0891- 18                1880         CLC
0892- 69 08             1890         ADC #8
0894- A8                1900         TAY
0895- E8                1910         INX             BUMP SCREEN IMAGE INDEX
0896- E0 10             1920         CPX #16
0898- 90 F3             1930         BCC .2          MORE CHARACTERS
089A- EE C5 08          1940         INC EPROM.ADR   BUMP TO NEXT LINE OF EPROM
089D- AD CC 08          1950         LDA SCREEN.ADR+1   +$400                DATA
08A0- 69 03             1960         ADC #3          (CARRY = 1)
08A2- 8D CC 08          1970         STA SCREEN.ADR+1
08A5- C6 00             1980         DEC CNT8        NEXT SCREEN LINE
08A7- D0 E0             1990         BNE .1          ...IF ANY
08A9- 60                2000         RTS             RETURN
                        2010 *-------------------------------
                        2020 *       REVERSE THE ORDER OF BITS 6-0 IN A-REG
                        2030 *       (CHANGE XABCDEFG TO 0GFEDCBA)
                        2040 *-------------------------------
                        2050 REVERSE.BITS
08AA- 4A                2060         LSR             REVERSE 7 BITS
08AB- 26 01             2070         ROL B           A=0XABCDEF  B=XXXXXXXG
08AD- 4A                2080         LSR
08AE- 26 01             2090         ROL B           A=00XABCDE  B=XXXXXXGF
08B0- 4A                2100         LSR
08B1- 26 01             2110         ROL B           A=000XABCD  B=XXXXXGFE
08B3- 4A                2120         LSR
08B4- 26 01             2130         ROL B           A=0000XABC  B=XXXXGFED
08B6- 4A                2140         LSR
08B7- 26 01             2150         ROL B           A=00000XAB  B=XXXGFEDC
08B9- 4A                2160         LSR
08BA- 26 01             2170         ROL B           A=000000XA  B=XXGFEDCB
08BC- 4A                2180         LSR
08BD- 26 01             2190         ROL B           A=0000000X  B=XGFEDCBA
08BF- A5 01             2200         LDA B
08C1- 29 7F             2210         AND #$7F        0GFEDCBA
08C3- 60                2220         RTS
                        2230 *-------------------------------
                        2240 *       PICK UP A BYTE OF EPROM DATA,
                        2250 *       REVERSE THE BITS, AND STORE
                        2260 *       IT ON THE SCREEN.
                        2270 *-------------------------------
                        2280 GET.PUT
08C4- B9 FF FF          2290         LDA $FFFF,Y
08C5-                   2300 EPROM.ADR .EQ *-2
08C7- 20 AA 08          2310         JSR REVERSE.BITS
08CA- 9D FF FF          2320         STA $FFFF,X
08CB-                   2330 SCREEN.ADR .EQ *-2
08CD- 60                2340         RTS
```

S-C Word Processor note.....Mike Laumer

We recently had one customer give us a great compliment on the S-C Word Processor. He has given up on WORDSTAR! He found that the S-C Word Processor can read and write large text files 20 times faster than WORDSTAR and that scrolling was much quicker. He can be in and out of the S-C Word Processor before WORDSTAR even lets him type a single key. The S-C Word Processor is also much less expensive than WORDSTAR and you don't have to buy a Z-80 card!

His only desire was to have an 80 column version of the Word Processor. However, that wouldn't be nearly so fast since SCWP re-writes the screen on every keystroke. I have noticed also that the 40 column display never causes me eye strain, but all the 80 column displays do.

QUICKTRACE

relocatable program traces and displays the actual machine operations, *while* it is running without interfering with those operations. Look at these *FEATURES:*

***Single-Step** mode displays the last instruction, next instruction, registers, flags, stack contents, and six user-definable memory locations.*

***Trace** mode gives a running display of the Single-Step information and can be made to stop upon encountering any of nine user-definable conditions.*

***Background** mode permits tracing with no display until it is desired. Debugged routines run at near normal speed until one of the stopping conditions is met, which causes the program to return to Single-Step.*

***QUICKTRACE** allows changes to the stack, registers, stopping conditions, addresses to be displayed, and output destinations for all this information. All this can be done in Single-Step mode while running.*

***Two optional display formats** can show a sequence of operations at once. Usually, the information is given in four lines at the bottom of the screen.*

***QUICKTRACE** is completely transparent to the program being traced. It will not interfere with the stack, program, or I/O.*

***QUICKTRACE** is relocatable to any free part of memory. Its output can be sent to any slot or to the screen.*

***QUICKTRACE** is completely compatible with programs using Applesoft and Integer BASICs, graphics, and DOS. (Time dependent DOS operations can be bypassed.) It will display the graphics on the screen while **QUICKTRACE** is alive.*

***QUICKTRACE** is a beautiful way to show the incredibly complex sequence of operations that a computer goes through in executing a program*

QUICKTRACE *$50*

Is a trademark of Anthro-Digital, Inc.

Written by John Rogers

See these programs at participating Computerland and other fine computer stores.

Anthro - Digital Software, Inc.
P.O. Box 1385 Pittsfield, MA 01202

Apple Chips.....Bob and Bill

You may recall that when Bill reviewed Apple][Circuit Description last month, he bemoaned the lack of a "Cross Reference", by board location, of all the Apple's ICs. Well Bob has worked out a couple of tables to fill that gap, and we'll be including those tables in future shipments of the book.

In the meantime, here's another sort of table, showing the locations and descriptions of all the chips in your Apple. This one is organized by chip number.

Chip	Board Location(s)	Chip Description
555	A13 B3	Timer
558	H13	3 Timers
741	K13	Op Amp
2316B	A5 (Rev 7,RFI)	ROM (character generator)
2513	A5 (Rev 0,1)	ROM (character generator)
4116	C3-10 D3-10 E3-10	RAM
6502	H6-9	Microprocessor
9316B	F3-11 (6 chips)	ROM (monitor and language)
74LS00	A2	4 2-input NAND
74LS02	A12 A14 B13 B14	4 2-input NOR
74LS04	C11	6 Inverters
74LS08	B11 H1	4 2-input AND
74LS11	B12	3 3-input AND
74LS20	D2	2 4-input NAND
74LS32	C14	4 2-input OR
74LS51	C13	AND3-NOR2, AND2-NOR2
74LS74	A11 B10 J13	2 Flip-Flops
74LS86	B2	4 2-input XOR
74LS138	F12 F13 H2 H12	3-by-8 Decoder
74LS139	E2 F2	2 3-by-4 Decoders
74LS151	A9	1-of-8 Selector
74LS153	C1 E11 E12 E13	2 1-of-4 selectors
74LS161	D11-14	Counter
74 166	A3	8-bit Shift Register
74LS174	B5 B8	6 Flip-Flops
74LS175	B1	4 Flip-Flops
74LS194	A10 B4 B9	4-bit Shift Register
74LS195	C2	4-bit Shift Register
74LS251	H14	1-of-8 Selector
74LS257	A8 B7 C12 J1	4 1-of-2 Selectors
74LS259	F14	8-bit Addressable Latch
74LS283	E14	4-bit Full Adder
74LS367	H3 H4 H5 (on some models)	6 Bus Drivers
8T97	H3 H4 H5 (on most models)	6 Bus Drivers
8T28	H10 H11 (on rev 0,1,7)	4 Bus Buffers
8304	H10 (on ref RFI)	8 Bus Buffers

S-C CAPTURE -- A Modem Program for the Word Processor.....

Jim Church

If you like to sign on to the The Source or CompuServe or some such system, you should get a copy of the S-C Word Processor. I like to receive the programs from CALL-A.P.P.L.E. magazine by modem and the S-C Word Processor really makes that easy.

What you do is quite simple. Just put a copy of B.SC.CAPTURE on the disk with the Word Processor. Then, whenever you want to capture a session with a remote system, you can choose D from the word processor menu and BLOAD B.SC.CAPTURE. After the routine is loaded, return to the main menu and choose L to load a sign-on file containing the commands necessary to dial the number you want to call. Here is a sample sign-on file, which I use to call up The Source.

```
!pr2
Q*367-6021        (The Q is a Control-Q)
!pr768
```

Now choose P from the menu, and your word processor will start dialing the phone! From here on you just operate the remote system as usual. The top line of the screen will show the address where characters are being stored, and the rest of the screen shows the text you are entering and receiving.

When you want to quit, just type a Control Z to hang up your phone and return to the word processor's main menu. Select E and you will see a copy of everything that transpired. Now you can edit the text however you want to, and save it all to your disk.

The !pr768 command above is intended to provide a hook for a user-written printer driver. It sets the output hook at $36-37 to $300. The next time the Word Processor tries to output a character, it wakes up the capture routine, which completely takes over until it is turned off with a Control Z. This is slightly abusing the !pr directive, so if you follow this example for other routines, be sure to have lines like 1570-1590 at the beginning of your routine, and exit to $803 at the end, so the Word Processor can reconnect itself correctly.

That's all there is to it. You could probably do a lot to "smarten up" this dumb terminal program. The way I have done it, it recognizes a Control Z from the keyboard and filters out incoming Control J's. That's all it does. Probably it should filter out Control G too, at the very least. My intention is to demonstrate the simple fact that the word processor is a very versatile creature.

This works, the way it is, with the D. C. Hayes Micromodem II in Slot 2. If your modem is in a different slot, just change line 1260 to show the correct slot number.

```
                    1000 *--------------------------------
                    1010 *
                    1020 *           S-C CAPTURE
                    1030 *
                    1040 *    A COMMUNICATIONS MODULE FOR
                    1050 *      THE S-C WORD PROCESSOR
                    1060 *
                    1070 *           BY JIM CHURCH
                    1080 *
                    1090 *--------------------------------
                    1100 * FULL DUPLEX CAPTURE PROGRAM
                    1110 * WORKS WITH MICROMODEM II
                    1120 * AND S-C WORD PROCESSOR
                    1130 *
                    1140 * GO INTO EDITOR W/EMPTY BUFFER
                    1150 * ENTER COMMANDS AS FOLLOW:
                    1160 *
                    1170 * !pr2
                    1180 * Q$367-6021    THE "Q" IS A CONTROL-Q
                    1190 * !pr768
                    1200 *
                    1210 * LEAVE EDITOR, CHOOSE P ON MENU
                    1220 *--------------------------------
                    1230         .OR $300
                    1240         .TF B.SC.CAPTURE
                    1250
0002-               1260 SLOT    .EQ $02
0020-               1270 SLOT16  .EQ SLOT*16
                    1280
0000-               1290 PTR     .EQ $00
0022-               1300 WNDTOP  .EQ $22
0024-               1310 CH      .EQ $24
                    1320
03EA-               1330 HOOK    .EQ $3EA
                    1340
2000-               1350 BUFFER  .EQ $2000
                    1360
C000-               1370 KEYBOARD   .EQ $C000
C010-               1380 STROBE     .EQ $C010
C0A5-               1390 MM.CR2     .EQ $C085+SLOT16
C0A6-               1400 MM.STATUS  .EQ $C086+SLOT16
C0A7-               1410 MM.DATA    .EQ $C087+SLOT16
                    1420
F941-               1430 PRNTAX  .EQ $F941
FB2F-               1440 INIT    .EQ $FB2F
FC22-               1450 VTAB    .EQ $FC22
FC24-               1460 VTABZ   .EQ $FC24
FC58-               1470 HOME    .EQ $FC58
FDF0-               1480 COUT1   .EQ $FDF0
FE89-               1490 SETKBD  .EQ $FE89
FE93-               1500 SETVID  .EQ $FE93
                    1510 *--------------------------------
                    1520 SC.CAPTURE
0300- 20 2F FB      1530         JSR INIT       FIX SCREEN
0303- 20 58 FC      1540         JSR HOME       CLEAR SCREEN
0306- A9 01         1550         LDA #1         RESERVE TOP LINE
0308- 85 22         1560         STA WNDTOP      FOR LOCATION COUNTER
030A- 20 93 FE      1570         JSR SETVID     PR#0
030D- 20 89 FE      1580         JSR SETKBD     IN#0
0310- 20 EA 03      1590         JSR HOOK       TELL DOS
0313- A2 00         1600         LDX #0         WORD PROCESSOR
0315- 8E 00 20      1610         STX BUFFER      NEEDS 0 AT $2000
0318- E8            1620         INX
0319- 86 00         1630         STX PTR        START POINTER
031B- A9 20         1640         LDA /BUFFER     AT $2001
031D- 85 01         1650         STA PTR+1
                    1660
                    1670 TERMINAL
031F- AD 00 C0      1680         LDA KEYBOARD   KEY DOWN?
0322- 10 15         1690         BPL MODEM      NO, CHECK MODEM
0324- 8D 10 C0      1700         STA STROBE     YES, CLEAR STROBE
0327- C9 9A         1710         CMP #$9A       CONTROL Z?
0329- F0 4C         1720         BEQ QUIT       YES, LEAVE
032B- 48            1730         PHA            SAVE KEYPRESS
032C- AD A6 C0      1740 .1      LDA MM.STATUS  CHECK IF THE TRANSMIT
032F- 29 02         1750         AND #$02        REGISTER EMPTY BIT IS SET
0331- F0 F9         1760         BEQ .1         NO, WAIT FOR IT
0333- 68            1770         PLA            YES, GET KEY BACK
0334- 8D A7 C0      1780         STA MM.DATA    SEND IT
0337- 30 E6         1790         BMI TERMINAL    AND LOOP AGAIN
                    1800
```

```
0339- AD A6 C0 1810 MODEM   LDA MM.STATUS CHECK IF THE RECEIVER
033C- 29 01    1820         AND #$01       REGISTER FULL BIT IS SET
033E- F0 DF    1830         BEQ TERMINAL NO, LOOP AGAIN
0340- AD A7 C0 1840         LDA MM.DATA  YES, GET CHARACTER
0343- 09 80    1850         ORA #$80     SET HI BIT
0345- C9 8A    1860         CMP #$8A     CONTROL J?
0347- F0 D6    1870         BEQ TERMINAL IGNORE IT
0349- 20 F0 FD 1880         JSR COUT1    PRINT CHAR
034C- A0 00    1890         LDY #0
034E- 91 00    1900         STA (PTR),Y  CAPTURE IT IN BUFFER
               1910
0350- E6 00    1920 INCR    INC PTR      BUMP POINTER LO
0352- D0 08    1930         BNE COUNT     NOT 0
0354- E6 01    1940         INC PTR+1    BUMP POINTER HI
0356- A5 01    1950         LDA PTR+1    CHECK IF
0358- C9 96    1960         CMP #$96      BUFFER END?
035A- B0 1B    1970         BCS QUIT     FULL BUFFER, LEAVE
               1980
035C- A5 24    1990 COUNT   LDA CH       SAVE CH
035E- 48       2000         PHA           ON STACK
035F- A9 00    2010         LDA #0       TOP LINE
0361- 20 24 FC 2020         JSR VTABZ     FOR LOCATION COUNTER
0364- A9 14    2030         LDA #$14     COL 20
0366- 85 24    2040         STA CH        IN CH
0368- A5 01    2050         LDA PTR+1    HI BYTE OF LOCATION
036A- A6 00    2060         LDX PTR      LO BYTE
036C- 20 41 F9 2070         JSR PRNTAX   PRINT ADDRESS
036F- 68       2080         PLA          GET OLD CH AND RETURN
0370- 85 24    2090         STA CH        TO WHERE WE WERE
0372- 20 22 FC 2100         JSR VTAB     OLD LINE
0375- 90 A8    2110         BCC TERMINAL START OVER
               2120
0377- A9 00    2130 QUIT    LDA #$00     END-OF-TEXT MARKER
0379- 91 00    2140         STA (PTR),Y   FOR WORD PROCESSOR
037B- A9 05    2150         LDA #$05     HANG UP PHONE
037D- 8D A5 C0 2160         STA MM.CR2    AT CONTROL REGISTER
0380- 4C 03 08 2170         JMP $803     COLDSTART WORD PROCESSOR
```

S-C Macro Cross Assemblers

The high cost of dedicated microprocessor development systems has forced many technical people to look for alternate methods to develop programs for the various popular microprocessors. Combining the versatile Apple II with the S-C Macro Assembler provides a cost effective and powerful development system. Hobbyists and engineers alike will find the friendly combination the easiest and best way to extend their skills to other microprocessors.

The S-C Macro Cross Assemblers are all identical in operation to the S-C Macro Assembler; only the language assembled is different. They are sold as upgrade packages to the S-C Macro Assembler. The S-C Macro Assembler, complete with 100-page reference manual, costs $80; once you have it, you may add as many Cross Assemblers as you wish at a nominal price. The following S-C Macro Cross Assembler versions are now available, or soon will be:

Motorola:	6800/6801/6802	now	$32.50
	6805	now	$32.50
	6809	now	$32.50
	68000	now	$50.00
Intel:	8048	now	$32.50
	8051	now	$32.50
	8085	soon	$32.50
Zilog:	Z-80	now	$32.50
RCA:	1802/1805	now	$32.50
Rockwell:	65C02	now	$20.00
DEC:	PDP-11/LSI-11	now	$50.00

The S-C Macro Assembler family is well known for its ease-of-use and powerful features. Thousands of users in over 30 countries and in every type of industry attest to its speed, dependablility, and user-friendliness. There are 20 assembler directives to provide powerful macros, conditional assembly, and flexible data generation. INCLUDE and TARGET FILE capabilities allow source programs to be as large as your disk space. The integrated, co-resident source program editor provides global search and replace, move, and edit. The EDIT command has 15 sub-commands combined with global selection.

Each S-C Assembler diskette contains two complete ready-to-run assemblers: one is for execution in the mother-board RAM; the other executes in a 16K RAM Card. The HELLO program offers menu selection to load the version you desire. The disks may be copied using any standard Apple disk copy program, and copies of the assembler may be BSAVEd on your working disks.

S-C Software Corporation has frequently been commended for outstanding support: competent telephone help, a monthly (by subscription) newsletter, continuing enhancements, and excellent upgrade policies.

S-C Software Corporation (214) 324-2050
P.O. Box 280300, Dallas, Texas, 75228

A PAUSE Directive.....Mike Laumer

Maybe your source code has outgrown even two disks and you need to know when to swap disks during assembly. Maybe you're using a single-sheet printer and need to change pages. Maybe you want to change typefaces on your letter-quality printer. Maybe you want to check the address of a routine or variable, without having to constantly watch the screen until it comes along. For whatever reason, you need to have the S-C Macro Assembler pause during assembly. Here is a new .US directive to let you do just that!

With this directive, you can insert a line like this anywhere in your code:

```
1300        .US SWAP SOURCE DISK
```

In each pass, when the assembler encounters this line it will pause, display "SWAP SOURCE DISK" in inverse text at the bottom of the screen, beep twice, and wait for a keypress. You can take whatever action you need to, and press any key to resume assembly.

The listing is for the Language Card version of the assembler. If you are using the main memory version, you don't need to worry about write-enabling and -protecting, so you can just delete lines 1220, 1230 and 1280.

The values for the .EQ statements in lines 1170-1180 depend on whether you are using the Main Memory or the Language Card assembler, and whether you have Version 1.0 or 1.1. Here's a table of the values for US.VCTR and SC.CMNT:

	Main Memory	Language Card	Version
US.VCTR	$100C	$D00C	Both
SC.CMNT	$1FD8	$E124	1.0
	$1FCA	$E0E4	1.1

That's all there is to it! Now you don't have to constantly stare at the screen during those long assemblies. Now you can sit back and wait for your Apple to call you when it needs you.

```
            1000 *--------------------------------
            1010 *   .US DIRECTIVE TO PAUSE DURING ASSEMBLY
            1020 *
            1030 *       SYNTAX:  .US <phrase>
            1040 *       RESULT:  Displays <phrase> in inverse text
            1050 *                and waits for a keypress
            1060 *
            1070 *--------------------------------
007B-       1080 CHR.PTR   .EQ $7B
0200-       1090 WBUF      .EQ $200
07D0-       1100 CORNER    .EQ $7D0
C000-       1110 KEYBOARD  .EQ $C000
C010-       1120 STROBE    .EQ $C010
C080-       1130 PROTECT   .EQ $C080
C083-       1140 ENABLE    .EQ $C083
FBE2-       1150 BELL      .EQ $FBE2
```

D O W N L O A D I N G
CUSTOM CHARACTER SETS

One of the features 'hidden' in many printers available today is their ability to accept user-defined character sets. With the proper software, these **custom characters** are 'downloaded' from your Apple II computer to the printer in a fraction of a second. Once the printer has 'learned' these new characters, they will be remembered until the printer is turned off.

After the downloading operation, you can use your printer with virtually any word processor. Just think of the possibilities! There's nothing like having your own **CUSTOM CHARACTERS** to help convey the message. And you still have access to those built-in fonts as well! **Here's a quick look at some possible variations:**

	BUILT-IN	CUSTOM
10CPI:	AaBbCcDdEeFfGgHhIiJjKk	**AaBbCcDdEeFfGgHhIiJjKk**
12CPI:	AaBbCcDdEeFfGgHhIiJjKk	**AaBbCcDdEeFfGgHhIiJjKk**
17CPI:	AaBbCcDdEeFfGgHhIiJjKk	**AaBbCcDdEeFfGgHhIiJjKk**
5CPI:	AaBbCcDdEeFf	**AaBbCcDdEeFf**
6CPI:	AaBbCcDdEeFf	**AaBbCcDdEeFf**
8CPI:	AaBbCcDdEeFf	**AaBbCcDdEeFf**

And let's not forget Enhanced and Underined printing as well...

AaBbCc**DdEeFfGgHh**IiJjKk	**AaBbCcDdEeFfGgHhIiJjKk**
AaBbCc<u>DdEeFfGgHh</u>IiJjKk	**AaBbCc<u>DdEeFfGgHh</u>IiJjKk**

The Font Downloader & Character Editor software package has been developed by RAK-WARE to help you unleash the power of your printer. The basic package includes the downloading software with 4 fonts to get you going. Also included is a character editor so that you can turn your creativity loose. Use it to generate unique character fonts, patterns, symbols and graphics. A detailed user's guide is provided on the program diskette.

SYSTEM REQUIREMENTS:
* APPLE II, APPLE II Plus, APPLE //e or lookalike with 48K RAM
* 'DUMB' Parallel Printer Interface Board (like Apple's Parallel Printer Interface, TYMAC's PPC-100 or equivalent)

The Font Downloader & Editor package is only $39.95 and is currently available for either the Apple Dot Matrix Printer or C.Itoh 8510AP (specify printer). Epson FX-80 and OkiData versions coming soon. Enclose payment with order to avoid $3.00 handling & postage charge.

R A K - W A R E
41 Ralph Road West Orange New Jersey 07052

Say You Saw It In **APPLE ASSEMBLY LINE**!

```
                        1160
D00C-                   1170 US.VCTR   .EQ $D00C
E124-                   1180 SC.CMNT   .EQ $E124
                        1190 *-------------------------------
                        1200         .OR $300
                        1210 *-------------------------------
0300- AD 83 C0          1220         LDA ENABLE        WRITE ENABLE
0303- AD 83 C0          1230         LDA ENABLE        RAM CARD
0306- A9 14             1240         LDA #PAUSE
0308- 8D 0D D0          1250         STA US.VCTR+1     POINT .US VECTOR
030B- A9 03             1260         LDA /PAUSE
030D- 8D 0E D0          1270         STA US.VCTR+2     TO PAUSE ROUTINE
0310- AD 80 C0          1280         LDA PROTECT       PROTECT CARD
0313- 60                1290         RTS
                        1300 *-------------------------------
0314- 20 E2 FB          1310 PAUSE    JSR BELL         BEEP
0317- A2 00             1320         LDX #0
0319- A4 7B             1330         LDY CHR.PTR      CHAR POINTER
031B- B9 00 02          1340 .1      LDA WBUF,Y       GET CHAR FROM CALL LINE
031E- F0 0B             1350         BEQ .2           END OF LINE?
0320- 29 3F             1360         AND #$3F         NO, INVERT CHAR
0322- 9D D0 07          1370         STA CORNER,X AND PUT IT AT BOTTOM OF
0325- E8                1380         INX                                SCREEN
0326- C8                1390         INY
0327- E0 28             1400         CPX #40          LINE FULL?
0329- 90 F0             1410         BCC .1           NO, GET ANOTHER CHAR
                        1420
032B- 20 E2 FB          1430 .2       JSR BELL         BEEP
032E- AD 00 C0          1440 .3       LDA KEYBOARD
0331- 10 FB             1450         BPL .3           WAIT FOR KEYPRESS
0333- 8D 10 C0          1460         STA STROBE
0336- 4C 24 E1          1470         JMP SC.CMNT      RETURN TO ASSEMBLY
                        1480 *-------------------------------
```

Some New Cards

1. Bob Stout just called from Houston to renew his subscription to AAL, and to tell me about a new toy he's getting. It seems that Legend Industries has a new kind of RAM card, containing 18K of static RAM, with battery backup.

16K of the memory on the card is mapped just like a language card, so it can be used in slot 0. The card also has a hardware write-protect switch, that you can throw to completely protect the memory. Once you have done that whatever you have stored in the card is there to stay.

The card can also be used in a higher slot for boot-up operation. The other 2K of memory is mapped at $CN00 and $C800, just like the ROM on a standard peripheral card. Think of the possibilities!

This new card from Legend is available with either NiCad or Lithium batteries. This gives you a choice between rechargeability or very long power-off life (about 2 years). The price is $149.95.

2. Saturn Systems has introduced a card with 64K RAM and a 6502 on it. The CPU runs at 3.6 MHz, compared to Apple's roughly 1 MHz. Comes with a pre-boot disk to let you use this faster processor with Applesoft, Pascal, and Integer BASIC. Price is $599. See their ad in the latest Softalk Magazine.

3. Analytical Engines, Inc. has one-upped the DTACK Grounded board. For only $1550, you can plug in an 8 MHZ 68000 card with 128K RAM (expandable to 512K on the card!). You can upgrade to a 12.5 MHz chip if you really need it. DTACK is NOT grounded on this board, so you have access to the full 16-megabyte address space. The 16K ROM on the board contains monitor functions and diagnostics. YOu can replace the ROM with up to 64K of EPROM if you want. Software? The price includes a complete UCSD P-system (I think he said version 4.1) with Pascal, Basic, and Fortran compilers. You also get an Applesoft-compatible BASIC interpreter that runs entirely inside the 68000. CP/M-68 is optional, and Unix is supposed to be available soon. See their ad in the latest Nibble Magazine.

4. Lee Meador has designed a board with 64K RAM, 4K EPROM, and a 2MHz 6502 on it. This unique board does not talk directly to the Apple bus; instead, there are two parallel ports (I presume implemented with a 6522 chip). One 8-bit port talks to the Apple I/O bus, and the other is available to outside devices. Software runs on the board at 2MHz, and at the same time your Apple chips do their 1MHz processing. I can think of a lot of neat ways to use Lee's board, including as a printer buffer/controller, as a high-speed math processor, as a hard disk interface, and so on. If enough of you are interested, Lee will sell these for around $500 each, along with some demonstration software.

FADD -- Find ADDress references.....Brooke Boering

Recently I have been messing around with modifications to DOS. Since I didn't have the complete source code for it, I simply used the explanations in "Beneath Apple DOS". I did find that I also needed a utility to locate all references to certain addresses. FADD was the result, and it's mighty useful. It's much quicker than doing a complete disassembly.

FADD will locate all instructions within 64K of memory referring to a given address. It skips the $C000-$CFFF (I/O) pages and avoids missing memory by doing a double read test.

It is intended to be used by the serious assembly language programmer for debugging and analysis. I'ts faster than doing a disassembly, though not quite so informative.

FADD is origined at $300 (what else?) and uses 8 zero page locations that are generally unused by programs except as scratch. You can alter both the origin point and zero page locations to suit your individual needs.

To use FADD:

1- BLOAD B.FADD
2- Get to Monitor
3- 'Fat finger' your address into 6-7 in HI-LO order.
4- Execute with a '300G'

NOTE: Use the spacebar to pause/release listing.

```
                    1000 ********************************
                    1010 *                              *
                    1020 *           F A D D            *
                    1030 *                              *
                    1040 *  ( FIND ADDRESS REFERENCES ) *
                    1050 *   ------------------------   *
                    1060 *                              *
                    1070 *    A PUBLIC DOMAIN UTILITY   *
                    1080 *                              *
                    1090 *     BY.. BROOKE W BOERING    *
                    1100 *                              *
                    1110 ********************************
                    1350
                    1360 * TO USE:
                    1370 * 1- BLOAD FADD.OBJ
                    1380 * 2- GET TO MONITOR
                    1390 * 3- 'FAT FINGER' YOUR ADDRESS
                    1400 *    INTO 6-7 IN HI-LO ORDER.
                    1410 *   NOTE ------>  <--------
                    1420 * 4- EXECUTE WITH A '300G'
                    1430
                    1460 *-----------------------------------
                    1470 *          E Q U A T E S
                    1480
0006-               1490 TARGHI .EQ $6
0007-               1500 TARGLO .EQ $7
                    1510 * NOTE: ABOVE REVERSES NORMAL LO/HI-BYTE
                    1520 *  ORDER FOR EASIER KEYIN FROM MONITOR
0008-               1530 WHER   .EQ $8
0008-               1540 WHERLO .EQ $8
0009-               1550 WHERHI .EQ $9
                    1560
002F-               1570 LENGTH  .EQ $2F
003A-               1580 PCL     .EQ $3A
003B-               1590 PCH     .EQ $3B
0030-               1600 COLOR   .EQ $30
                    1610
F88E-               1620 INSDS2  .EQ $F88E
F8D0-               1630 INSTDSP .EQ $F8D0
F956-               1640 PCADJ3  .EQ $F956
FD8E-               1650 CROUT   .EQ $FD8E
                    1660 *-----------------------------------
                    1670        .OR $300
                    1680 *      .TF B.FADD
                    1690 *-----------------------------------
                    1700 START
                    1710
0300- A2 00         1720  LDX #0
0302- 86 08         1730  STX WHERLO    START AT BEGINNING
0304- 86 09         1740  STX WHERHI    OF MEMORY
                    1750
                    1760 *-- CHECK FOR DIRECT REFERENCE
                    1770 .1
0306- A0 00         1780  LDY #0
0308- B1 08         1790  LDA (WHER),Y GET WHERAT-LO
030A- 85 30         1800  STA COLOR    SAVE TEMP
030C- B1 08         1810  LDA (WHER),Y GET IT AGAIN
030E- C5 30         1820  CMP COLOR    STILL THE SAME?
0310- D0 5D         1830  BNE .8       NO, SKIP IT, NO MEMORY HERE
                    1840 * (FALL THROUGH IF MEMORY AT THIS ADDRESS)
                    1850
0312- C5 07         1860  CMP TARGLO   ? TARGET-LO ?
0314- D0 20         1870  BNE .3       NO, GO AHEAD
0316- C8            1880  INY
0317- B1 08         1890  LDA (WHER),Y GET WHERAT-HI
0319- C5 06         1900  CMP TARGHI   ? TARGET-HI ?
031B- D0 19         1910  BNE .3       NO, GO AHEAD
                    1920 * (FALL THROUGH IF 2-BYTE MATCH ON TARGET)
                    1930
                    1940 *-- APPARENT MATCH;
                    1950 *     MAKE SURE IT'S A 3-BYTE INSTRUCTION
                    1960 .2
031D- A4 09         1970  LDY WHERHI   GET ADDRESS OF MATCH
031F- A6 08         1980  LDX WHERLO
0321- D0 01         1990  BNE .24
0323- 88            2000  DEY          POINT TO INSTRUCTION BYTE
```

```
                        2010 .24
0324- CA                2020  DEX
0325- 86 3A             2030  STX PCL       AND SET PROGRAM COUNTER
0327- 84 3B             2040  STY PCH
                        2050
0329- A2 00             2060  LDX #0
032B- A1 3A             2070  LDA (PCL,X)   GET OPCODE
032D- 20 8E F8          2080  JSR INSDS2    USE MONITOR DISASSEMBLER ROUTINE
0330- A5 2F             2090  LDA LENGTH
0332- C9 02             2100  CMP #2        3-BYTE INSTRUCTION?
0334- F0 26             2110  BEQ .6        OK; GO AHEAD TO DISPLAY
                        2120 * (FALL THROUGH WHEN NOT A 3-BYTE INSTR)
                        2130
                        2140 *-- CHECK FOR RELATIVE BRANCH
                        2150 .3
0336- A0 00             2160  LDY #0
0338- B1 08             2170  LDA (WHER),Y GET INSTRUCTION BYTE
033A- 29 1F             2180  AND #$1F     ISOLATE SIGNIFICANT BITS
033C- C9 10             2190  CMP #$10      A BRANCH INSTRUCTION?
033E- D0 2F             2200  BNE .8        DEFINITELY NOT
                        2210 * (FALL THROUGH WHEN A BRANCH INSTRUCTION)
                        2220
                        2230 *-- TEST IF BRANCHING TO TARGET
                        2240 *     NOTE: USING MONITOR TECHNIQUE
                        2250 .4
0340- A6 08             2260  LDX WHERLO    PRESET FOR PCADJ3
0342- A4 09             2270  LDY WHERHI
0344- 86 3A             2280  STX PCL       SET PC TO OPCODE BYTE
0346- 84 3B             2290  STY PCH
0348- A0 01             2300  LDY #1
034A- B1 08             2310  LDA (WHER),Y GET OFFSET BYTE
034C- 20 56 F9          2320  JSR PCADJ3    LEAVES EFFECTIVE ADDRESS-1
                        2330 *              IN Y AND A
034F- AA                2340  TAX
0350- E8                2350  INX
0351- D0 01             2360  BNE .43
0353- C8                2370  INY
                        2380 .43
                        2390 *-- NOW 'BRANCH TO' ADDRESS IS IN Y AND X
0354- E4 07             2400  CPX TARGLO
0356- D0 17             2410  BNE .8
0358- C4 06             2420  CPY TARGHI
035A- D0 13             2430  BNE .8
                        2440 * (FALL THROUGH ON MATCH)
                        2450
                        2460 *-- DISPLAY MATCHED INSTRUCTION
                        2470 .6
                        2480 * PCL/PCH ARE SET
035C- 20 D0 F8          2490  JSR INSTDSP  <= MONITOR ROUTINE
                        2500
                        2510 *-- ALLOW KEYED PAUSE/RELEASE
                        2520 .7
035F- 2C 00 C0          2530  BIT $C000     KEY DOWN?
0362- 10 0B             2540  BPL .8        NO, GO AHEAD
0364- 2C 10 C0          2550  BIT $C010     YES, CLEAR STROBE
                        2560 .77
0367- 2C 00 C0          2570  BIT $C000     RELEASED?
036A- 10 FB             2580  BPL .77       NO, LOOP TIL SO
036C- 2C 10 C0          2590  BIT $C010     YES, CLEAR STROBE
                        2600
                        2610 *-- POST DISPLAY (OR NO MATCH)
                        2620 .8
036F- E6 08             2630  INC WHERLO    KICK ADDRESS
0371- D0 93             2640  BNE .1        LOOP 255 OF 256
0373- E6 09             2650  INC WHERHI    KICK ADDR PAGE#
0375- F0 0C             2660  BEQ .9        EXIT AT 65536 OVFLO
                        2670
                        2680 *-- AT NEW PAGE !!
0377- A5 09             2690  LDA WHERHI
0379- C9 C0             2700  CMP #$C0      AT THE I/O PORTS ?
037B- D0 89             2710  BNE .1        NO, LOOP BACK
037D- A9 D0             2720  LDA #$D0      YES, SKIP 'EM
037F- 85 09             2730  STA WHERHI    : (AVOID PROBLEMS)
0381- D0 83             2740  BNE .1        LOOP BACK
                        2750
                        2760 .9
0383- 4C 8E FD          2770  JMP CROUT     RETURN THROUGH CROUT
```

Generating Parity.....Bob Sander-Cederlof

When large amounts of data are being moved around it is easy to garble some. When you transmit characters over the telephone, or read them from a tape or disk, you want some kind of assurance that the message does not get modified by the medium.

Lots of schemes have been invented to prevent, detect, and even correct transmission errors: checksums, parity, cyclic redundancy codes, and more.

Checksums are used inside the Apple all the time. If you ever used cassette tapes with your Apple, you were re-assured to know that each program was recorded with a checksum. DOS 3.3 adds a checksum to the end of every sector on the disk. The checksum is re-computed when you read a tape or disk sector; if the result is different, at least one bit in the data is wrong.

Most of the checksums I have seen are of the exclusive-or type. All the bytes in the data record are EORed together, and the result is written at the end of the record. When the data is read, the in-coming bytes are again EORed together, and finally EORed with the checksum itself. If the final result is non-zero, an error occurred.

Checksums in the Apple are usually one byte wide. However, for more security, you could form a wider checksum. Or you could ADD the bytes together and store a two byte sum. Or store the complement of the sum, so that adding all the bytes plus the complement will give a zero result if there are no errors. [Checksums may check out OK even though errors occur, if the errors are sneaky enough to cancel each other out.]

Parity is really a kind of checksum, but only one bit wide. A series of bits is EORed together, and the single-bit result is the parity value. In an ASCII character there is provision for the leading bit position to be used for storing a parity bit. An eight-bit byte holds seven data bits plus a parity bit.

There are two kinds of parity in use: even and odd. Even parity makes the total number of 1-bits in the stream of bits even; odd parity makes the total number of 1-bits odd. Both even and odd are in use today in various kinds of equipment. Many terminals and serial communication boards allow you to select even, odd, or no parity. Looking at the ASCII code for a couple of letters, each could be transmitted in four ways:

```
Letter "M"  0 1 0 0 1 1 0 1 -- No parity, 8th bit always 0
            1 1 0 0 1 1 0 1 -- No parity, 8th bit always 1
            0 1 0 0 1 1 0 1 -- Even parity
            1 1 0 0 1 1 0 1 -- Odd parity

Letter "Q"  0 1 0 1 0 0 0 1 -- No parity, 8th bit always 0
            1 1 0 1 0 0 0 1 -- No parity, 8th bit always 1
            1 1 0 1 0 0 0 1 -- Even parity
            0 1 0 1 0 0 0 1 -- Odd parity
```

Sometimes I have needed a quick way to generate or verify a parity bit with software. These matters are usually handled in hardware, but not always.

In the 6502, it is a very simple matter to rotate a byte around and count the number of one bits present. Then the parity bit can be merged with the byte, or compared with what is already there.

The following subroutine (PARITY) computes the parity bit and merges it with the data byte. Call PARITY with the character to be merged in the A-register. Only the seven data bits will be counted. As written, the subroutine computes an odd parity bit. You can change line 1030 to "LDX #0" to compute even parity.

```
1000 *-------------------------------
1010 *   Compute and merge parity bit
1020 *-------------------------------
1030 PARITY LDX #1      (#0 FOR EVEN PARITY)
1040        ASL         SHIFT PARITY POSITION OUT
1050        PHA         SAVE SHIFTED CHARACTER
1060 .1     BPL .2      IF NEXT BIT = 0, DON'T COUNT
1070        INX         IF NEXT BIT = 1, COUNT IT
1080 .2     ASL         SHIFT IN NEXT BIT
1090        BNE .1      IF ANY REMAINING BITS = 1
1100        TXA         GET COUNT OF 1-BITS
1110        LSR         EVEN/ODD BIT OF COUNT INTO CARRY
1120        PLA         ORIGINAL CHAR BUT SHIFTED
1130        ROR         SHIFT PARITY BIT INTO BIT 8
1140        RTS
```

I wrote a little program to drive the PARITY subroutine, using all possible values from 0 through 127, and print out the results:

```
1150 *-------------------------------
1160 PRHEX  .EQ $FDDA  MONITOR PRINT (A) IN HEX
1170 COUT   .EQ $FDED  MONITOR PRINT CHAR IN (A)
1180 *-------------------------------
1190 DEMO   LDA #0     FOR CHAR = $00 TO $7F
1200        STA CHAR
1210 .1     JSR PARITY CALL THE PARITY SUBROUTINE
1220        JSR PRHEX  PRINT THE CHARACTER
1230        INC CHAR   NEXT CHAR
1240        LDA CHAR   SEE IF TIME FOR A NEW LINT
1250        AND #$07
1260        BEQ .2     YES
1270        LDA #$A0   <SPACE>
1280        BNE .3     ...ALWAYS
1290 .2     LDA #$8D   <RETURN>
1300 .3     JSR COUT   PRINT SPACE OR RETURN
1310        LDA CHAR
1320        BPL .1     STILL LESS THAN $80
1330        RTS        DEMO FINISHED
```

When I set it up for odd parity, here is part of the table printed out by DEMO:

```
80 01 02 83 04 85 86 07
08 89 8A 0B 8C 0D 0E 8F
10 91 92 13 94 15 16 97
   .
   .
F8 79 7A FB 7C FD FE 7F
```

Now, how about a subroutine to check parity? Here is a version that checks an 8-bit value for odd parity. Simply change line 1420 to "LDX #0" to check for even parity instead. The subroutine returns with CARRY CLEAR for good parity, or CARRY SET for bad parity.

```
1400 *-------------------------
1410 CHECK.PARITY
1420         LDX #1      (OR #0 FOR EVEN PARITY)
1430         PHA         SAVE ORIGINAL CHAR
1440 .1      ASL         SHIFT NEXT BIT INTO CARRY
1450         BEQ .2      NO REMAINING 1-BITS
1460         BCC .1      LEADING BIT NOT 1-BIT
1470         INX         COUNT THE 1-BIT
1480         BCS .1      ...ALWAYS
1490 .2      BCC .3      LATEST SHIFTED BIT WAS 0
1500         INX         LATEST SHIFTED BIT WAS 1
1510 .3      TXA         BIT COUNT
1520         LSR         SHIFT EVEN/ODD BIT INTO CARRY
1530         PLA         RESTORE ORIGINAL CHARACTER
1540         RTS
```

ROGRAM TOO LARGE???.....Lee Meador

I was writing an ampersand file-handling routine, using the File Manager in DOS as described in chapter 6 of Beneath Apple DOS. I wanted my Applesoft program to be able to set ONERR and catch errors in files (wrong name, too short, etc.) But I also wanted the error messages to come out in immediate mode or with no ONERR set. Since I was doing my own file-handling, I was going to have to provide my own error outputs. Originally I tried this:

```
ERROR  LDY #$0A        error code offset
       LDA ($04),Y     $04 -> FM parmlist
       JMP $A6D2       jump into DOS error handler
```

This worked OK when used in code that was called from an Applesoft program, but when I called it in immediate mode (from the "]") I would always get "ROGRAM TOO LARGE" when an error occurred.

You might guess that I have found a solution. The problem is caused when we jump into DOS at $A6D2 with the IO hooks still pointing into DOS. The routine starting at $A6D2 saves the error code in a temporary location at $AA5C and calls $A702 to print a "<return><beep><return>". Since we entered illegally that output goes to DOS at $9EBD, then via a JSR to $9ED1 where the accumulator is saved in a temporary location -- $AA5C! This leaves that last <return> in $AA5C.

When control returns to the error handler DOS then tries to look up error message number 141 ($8D) in the 16-entry table of offsets starting at $AA3F. This loads the offset from location $AACC, which happens to contain the high-order byte of the address of the OPEN command handler ($AB22)! This leaves the error message printer with an offset of $AB into the messages at $A971. And that is what produces "ROGRAM TOO LARGE". Look at the routines at $A6D2 and $A702 for more detail. $A702 is meant to be called with the error code in the X register.

Now here's a method to have the error handled correctly:

```
OUT.HOOK   .EQ $36
HARD.COUT  .EQ $FDF0

ERROR  LDA #HARD.COUT  point hook out of DOS
       STA OUT.HOOK
       LDA /HARD.COUT
       STA OUT.HOOK+1  DOS will fix it back
       LDY #$0A        index to error code
       LDA ($04),Y     $04 -> FM parmlist
       JMP $A6D2       do it ...
```

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $15 per year in the USA, sent Bulk Mail; add $3 for First Class postage in USA, Canada, and Mexico; add $13 postage for other countries. Back issues are available for $1.50 each (other countries add $1 per back issue for postage).